COMP5710 final review

CONFIGS

Two kinds of configuration:

1. Classic bash script approach
2. Infrastructure as code with snippet code

Languages:

1. Ansible
2. Puppet
3. Chef
4. Kubernates

“Configuration management ... is the discipline of identifying the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration and maintaining the integrity and traceability of the configuration throughout the system life cycle.”

It is concerned with managing evolving software systems

Misconfigurations can be very bad and some examples are:

- default configurations
- open cloud storage
- insure HTTP requests
- verbose error messages

CMMI

Software process maturity is the measure of long-term effectives of software engineering process.

Principal ideas:

1. Key elements of a mature process
2. Describe. How to make an immature process more disciplined mature process
3. Has 5 maturity levels, first has 5 common feures to specify key practices
    a. Initial
    b. Repeatable
    c. Defined
    d. Quantitively Managed
    e. Optimizing

SCRUM

Process: Vision > product backlog > spring planning > spring backlog > daily scrum > spring review > retrospective

Product owner:

1. Collects and organizes requirements
2. Prioritizes scope
3. Clarifies requirements
4. Accepts/rejects the implemented user stories

Developers:

1. Delivers product at the end of sprint
2. Cross-functional, works all aspects
3. Self-organizes and self-manages
4. Review work with PO

Scrum Master:

1. Coaches scrum
2. Helps team in all aspects

Can utilize Mob programming
Driver-navigator-pair programming

WATERFALL

The oldest and widest used process model
System engineering > requirements analysis > Design > implementation > testing > maintenance

BDD

Connecting BDD and SCRUM

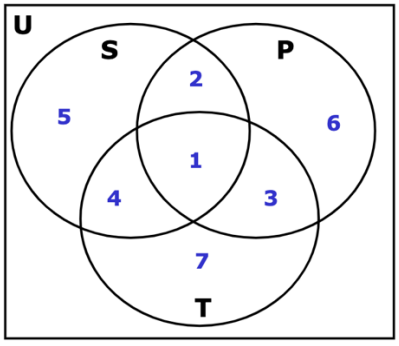
Product backlog items:
•User stories (functional requirements) (features of a new wanted capability)
•Quality attribute requirements
•Issues and bugs
•Technical issues (refactoring, introducing CI, etc.)

TESTING

Why is testing hard?:
The entire testing suite can be very overwhelming
It is very detailed
It is time consuming
Requires technical sophistication
Should be treated as craft
Requires upfront planning

Thinking About Testing
• Phase 0: Testing = Debugging
• Phase 1: Testing is an act whose purpose is to show that the software works.
• Phase 2: Testing is an act whose purpose is to show that the software does not work.
• Phase 3: Testing is an act whose purpose is not to prove anything, but to reduce the perceived risk of failure to an acceptable level.
• Phase 4: Testing is not an act; rather, it is a mindset that involves development and coding practices along with a systematic approach to exercising the software.

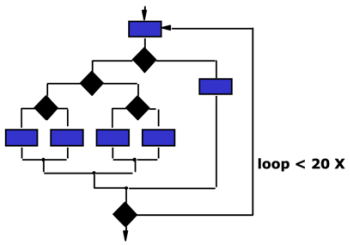
Test cases, specs, programmed behaviors:
S = Specified behaviors
P = Programmed behaviors
T = Tested behavior
U = All possible behaviors



Test case design:
Objective is to uncover errors, the criteria is in a complete manner, the constraint is in minimal effort and time

Testing Principles
• Tests should be traceable to requirements.
• Tests should be planned early.
• The Pareto principle applies.
• Testing should start small and then ramp up.
• You can't test everything.
• Testing should be done by an independent party.

Exhaustive Testing



There are 10^{14} possible paths. If we execute one test per millisecond, it would take 3,170 years to test this program.

Exhaustive testing is testing everything, selective testing is testing one path only
Should be done independently

Elements of Testability
• Operability—it operates cleanly
• Observability—the results of each test case are readily observed
• Controllability—the degree to which testing can be automated and optimized
• Decomposability—testing can be targeted
• Simplicity—reduce complex architecture and logic to simplify tests
• Stability—few changes are requested during testing
• Understandability—of the design

Testing Methods
• Structural (White Box) Testing
– Knowing the internal workings of a program, tests can be conducted to assure that the internal operation performs according to specification, and all internal components have been exercised.
– Test cases are based on internal structure of the program and a specific level of coverage.
• Functional (Black Box) Testing
– Knowing the specified functions that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.
– Test cases are based on external behavior as well as
internal structure.

TESTING METRIC

Software quality metrics, No matter the taxonomy or method of measurement, no real measurement of quality ever occurs; only surrogates can ever be measured.

A Few Measures and Metrics
- Lines of code (LOC)
  - Positives: can be counted
  - Negative: what to count
- Function Points (FP)
- Reliability Metrics
- Complexity Metrics
  - – Halstead Metrics (Linguistic metric)
  - – McCabe Metrics
  - – Complexity Profile Graph

TESTING USING GRAPHS

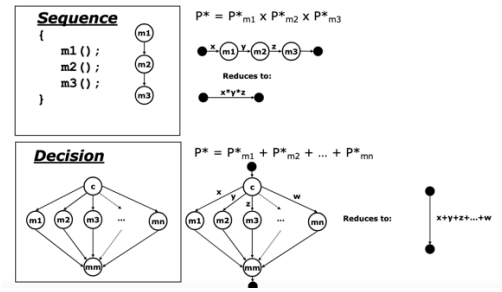
Nodes and edges (starting and terminal node/edge)
Paths and semi-paths (from-to node/edge)
Connected is 3 types: 1-connected, 2-connected, 3-connected
Graphs can be connected or strongly-connected
Different graphs for:
1. Sequence
2. If-then
3. If-then-else
4. Switch/multiway
5. Pre-test loop
6. Post-test loop

+ different programs

STRUCTURAL TESTING

Structural testing is based on selecting paths
through a code segment for execution.
Assume there is only one entry and exit point
The more paths are tested the more thoroughly it is tested

Path Arithmetic



Levels of Test Coverage
• Statement Coverage
– Sufficient number of test cases so that each statement is executed at least once
• Branch Coverage
– Sufficient number of test cases so that each branch of each decision is executed at least once
• Condition Coverage
– Sufficient number of test cases so that each condition (in each decision) takes on all possible outcomes at least once
• Decision/Condition Coverage
– Sufficient number of test cases so that each condition and each decision takes on all outcomes at least once

- All paths coverage (infeasible and impossible)
- Independent path/basis set coverage

Cyclomatic Complexity and IP Coverage:
V(G) Maximum number of required independent paths in a basis set
Construct by doing the shortest length and keep adding edges until all is covered

Test loops for:
• Looping to test for
– Initialization errors
– Index or incrementing errors
– Bounding errors which occur at loop limits
• Classes of loops
– Simple loop
– Nested loop
– Concatenated loop
– Unstructured loop

Data Flow Testing

• A form of structural testing that is based not on control flow, but on how data flows through the program.

• Focuses on the points at which variables receive values and the points at which these values are used.

• Two major flavors: DU-paths and Program slices.

COMBINATORIAL TESTING